Development and Performance of a Scalable Version of a Nonhydrostatic Atmospheric Model

Arthur A. Mirin and Gayle A. Sugiyama
Atmospheric Science Division,
Lawrence Livermore National Laboratory, Livermore, CA

Sue Chen, Richard M. Hodur, Teddy R. Holt, and Jerome M. Schmidt
Marine Meteorology Division,
Naval Research Laboratory, Monterey, CA

Abstract

The atmospheric forecast model of the Naval Research Laboratory's (NRL) Coupled Ocean/Atmosphere Mesoscale Prediction System (COAMPS) has been developed into a parallel, scalable model in a joint collaborative effort with Lawrence Livermore National Laboratory (LLNL). The new version of COAMPS has become the standard model of use at NRL and in LLNL's Atmospheric Science Division. The main purpose of this enterprise has been to take advantage of emerging scalable technology, to treat finer spatial and temporal resolutions needed in complex topographical or atmospheric conditions, as well as to allow the utilization of improved but computationally expensive physics packages. The parallel implementation facilitates the ability to provide real-time, high-resolution, multi-day numerical weather predictions for forecaster guidance, input to atmospheric dispersion simulations, and forecast ensembles.

Introduction

The three-dimensional Coupled Ocean/Atmosphere Mesoscale Prediction System (COAMPS) originally developed by the Naval Research Laboratory (NRL) for serial applications has been developed into a parallel, scalable model in a joint collaborative effort with Lawrence Livermore National Laboratory (LLNL). COAMPS consists of atmospheric and ocean data assimilation (including data quality control), analysis, initialization, and a nonhydrostatic atmospheric forecast model coupled to a hydrostatic ocean model (Hodur 1997). The atmospheric system has been used for operational mesoscale forecasting since 1996. It has provided products to the meteorological community from both a supercomputer central site (Fleet Numerical Meteorology and Oceanography Center located in Monterey, CA) and the Department of Energy's National Atmospheric Release Advisory Capability (NARAC) at LLNL. Regional sites using workstations (Naval centers, Universities, and Government Agencies) have recently begun to use COAMPS in an operational mode. In the research community, COAMPS has been used extensively for both idealized as well as real data simulations (Haack and Burk 2001, Haack et al 2001, Doyle and Shapiro 2000, Doyle et al. 2000, Dorman et al. 2000, Liu et al. 2000, Burk and Haack 2000, Westphal et al. 1999).

COAMPS description

COAMPS consists of an atmospheric data assimilation system containing a nonhydrostatic atmospheric forecast model and an ocean analysis and forecast model. The initial focus of the parallelization effort has been on the atmospheric forecast model. COAMPS solves the nonhydrostatic, compressible form of the dynamical equations, and includes relevant physical processes such as explicit moist physics, cumulus convection, and radiation, as well as parameterizations for subgrid-scale mixing. The vertical sigma coordinate is chosen to allow flow over an irregular surface and a variety of horizontal coordinate systems can be invoked. The equations are discretized using finite differences on an Arakawa C grid. The difference scheme is fundamentally explicit in the horizontal direction, with subcycling to evolve the faster moving sound and gravity waves. Some of the vertical phenomena are integrated implicitly. Most terms are represented to second order accuracy, with options to use fourth order methods for the diffusion and advection.

Programming practices

The underlying principle in COAMPS programming is that the code be capable of executing efficiently across vector, parallel, or symmetric multi-processor (SMP) machines by simply changing run-time options. This necessitates slightly more overhead through additional arrays and syntax logic, but pays dividends in flexibility and portability. The code, originally written in Fortran-77, now invokes a number of Fortran-90 constructs. Memory management is carried out using pointers and allocatable arrays. Representation of physical quantities on multiple grids is facilitated using derived data types. These derived types are used to maximize message lengths by taking horizontally-mapped data and creating new data types that include all vertical levels. Every effort is made to adhere to standards in order to achieve portability and minimize the recoding

needed to accommodate new architectures, and is in effect a form of optimization. Three specific areas related to programming practices will be discussed: 1) domain decomposition, 2) nesting, and 3) input/output (I/O). The differences between practices adopted for the new scalable code and the vector (or serial) code will be highlighted.

Domain decomposition

The integral design component of the new scalable COAMPS code is the use of horizontal domain decomposition. The decomposition of the entire model domain into subdomains is based upon run-time, user-specified values of the number of subdomains in the x- and y-directions. Based upon these values the decomposition technique automatically partitions the data among the nodes of a parallel machine into as equal a number of grid points per subdomain as possible for each x- or y-direction. (Because of the tight vertical coupling that exists in COAMPS, the decomposition is limited to the horizontal plane.) The user is also allowed run-time flexibility in specifying the number of grid points to be used in the halo region (usually either one or two grid points). The halo region is necessary to facilitate finite differencing and data communication with nearest neighbors. Because of fourth-order differencing in COAMPS, two halo points are typically used. Communication between subdomain processes is achieved using either Message Passing Interface (MPI) 1 or 2. Within each subdomain an additional level of parallelization is provided through the use of the de-facto standard OpenMP. OpenMP directives can also be executed for SMP-only applications.

Figure 1 shows a sample single nest COAMPS horizontal domain illustrating the relationship between decomposition and halo regions. The sample domain is dimensioned (1:m,1:n) (or 23 x 23) grid points in the x- and y-directions, as indicated by the heavy solid box. The domain is decomposed into nine subdomains (P0 to P8) in a 3 x 3 configuration. The heavy shaded areas indicate the computational region for each subdomain which extend from indices (imin:imax, jmin:jmax). These indices are those automatically computed to evenly distribute the data horizontally across subdomains. In the serial code this extent is analogous to (2:m-1, 2:n-1). Thus, no computations are performed on the outer-most row or column of data except for the specification of the lateral boundary condition data.

For this sample domain there are two halo points for each subdomain, illustrated using only subdomain 7 (P7) in Figure 1. The open circles represent grid points within the computational domain of P7. The two solid grid points surrounding the computational points represent the halo region for P7. The extent of P7 with two halo points is from (iminf:imaxf, jminf:jmaxf). Thus, indices for the halo regions are trivially computed as iminf=imax-nb, or imaxf=imax+nb, where nb is the number of halo points. Note that the halo points extend nb points into the neighboring subdomains.

Additional indices and variables are computed to expedite communications between processors. For example, each processor computes its “physical” extent (which can include points out to the boundary of the entire domain), x- and y-direction data widths, neighboring processor relationships (i.e., is this processor northernmost? southernmost?), etc. Indices and variables differ based upon whether the user chooses the run-time option for scalable or serial code (or a combination). For

serial applications long vector lengths are desired. However, for scalable code two-dimensional loop indices are limited by the horizontal extent of each subdomain. The logic of the new code allows for both types. This is achieved by specifying loop indices that collapse two-dimensional loops into a single dimension to take advantage of vectorization for serial applications, but remain as two-dimensional indices for scalable applications.

Nesting

An arbitrary number of inner nests can be specified in COAMPS. The only constraint is that the number of grid points be a multiple of three plus one with a consequent 3:1 reduction in horizontal resolution. There is a similar reduction in time step such that the inner (or child) nest is called three times for each parent time step. Each child nest is decomposed in a similar fashion as described in the section above. The added complexity arises from the required communications between the parent and child nests. Figure 2 shows the single nest of Figure 1 with a child nest included. The open circles show the computational area of parent subdomain P0 and the solid circles the two halo points. The heavy solid line through the child nest illustrates the extent of the parent subdomain into the child. Note that because of the 3:1 restriction in nest resolution, parent points coincide with every third child point.

For this sample nine subdomain example, the child nest is similarly decomposed with the same 3 x 3 configuration (Figure 3), though COAMPS does allow different configurations for each nest as long as the total number of subdomains is the same. The child nest is dimensioned (25 x 25 grid points) with the two solid grid points surrounding the computational area of child subdomain 3 (C3) indicating the halo region. The heavy solid lines through C1, C3, and C4 represent the boundaries of parent subdomain P0 (as shown in Figure 2). For this simple example, the other parent subdomains are not shown. Note however that any given child subdomain can overlap with several different parent subdomains.

For one-way interactive nesting, the communication of boundary information is from the parent nests to the child nests at every time step of the parent nest. The boundary information from the parent nest is communicated to the child over a user-specified region surrounding the entire child domain called the blend zone (Figure 3). This blend zone is typically five to seven child grid points wide on the child nest. The programming practice used in the nesting strategy to distinguish regions of the appropriate parent and child nests that need to be communicated is to employ masks. Masks are simply “on or off” switches. Data points for a given subdomain, be it parent or child, are assigned mask values of 1 (“on”) if they are in the blend zone and 0 (“off”) if they are not. Masks for each child subdomain are trivially determined from the number of blend zone points on the perimeter of the subdomain. Parent masks are determined based upon the origin reference location and extent of the inner nests relative to the parent nest. Once the masks have been determined, each parent subdomain determines, based upon the number of “on” mask values and their relationship to child subdomains, whether and where it should send data. Similarly, child subdomains determine from which parent subdomain they should receive based upon their mask values and their relationship to the parent subdomains.

The data received from a parent processor is stored in a temporary array that resides on each child processor. Each temporary array (or envelope) is defined in parent coordinate space and is designed to just cover the domain size of the entire child subdomain. Only those parent points required to compute the horizontal interpolation within the blend zone on the nest boundaries are sent to the envelope array. Offsets are computed to map the arriving parent grid points into the appropriate location within the envelope. Once the parent data points arrive, horizontal interpolation from the coarse mesh data of the parent grid to the fine mesh points of the child grid is then performed using the previously defined masks. As illustrated in Figure 4, the position and size of the envelope is required to be slightly different for interpolating data to the mass and momentum points due to the Arakawa-C grid staggering.

The hatched region in Figure 5 shows the parent envelope for mass points that must be communicated from parent P0 to child C3 for the sample horizontal domain. The region contains nine parent grid points that represent the intersection of P0 and the child blend zone on C3. Similar regions would exist for each blend zone region on all child subdomains except C4. C4 is an interior subdomain and does not have an exterior blend zone region.

COAMPS also has the option to allow the inner nests to move in time. This adds another level of complexity to the programming because additional communications and arrays are needed to record the movement of the nests. The strategy employed follows the basic programming practices for a fixed nest discussed above. The communication of boundary information is similar except that at the time a nest is moved, the reference points for the parent grid must be reset to allow for the correct information to be communicated to the child nests. In addition, when a child nest is moved, the new region in which the nest has moved must be interpolated from existing information on the parent nest. The interpolated region could span across a variety of parent processors and require additional communications.

Input/Output (I/O)

COAMPS processes a tremendous amount of data for any given analysis and forecast cycle. Thus, efficient handling of the input and output of this data is crucial for good computational performance. The input of initial and boundary condition data into COAMPS is handled using either MPI-1 or MPI-2 constructs as specified by the user at run-time. For either option COAMPS uses a designated I/O task that first reads the entire two- or three-dimensional field into its own local memory. Using either MPI-1 or MPI-2, the data are then communicated to the appropriate subdomains as specified by the domain decomposition.

The output of COAMPS data can be handled in two separate ways as specified by the user at run-time. The first option is to use a designated I/O task similar to the input procedure described above. The I/O task collects data from all subdomains and writes to the designated output file. The other option is to use MPI-2 I/O directives. In this option each task writes into the designated output file only the portion of the data that it contains, using the appropriate location offset. Scaling results for real-data COAMPS simulations with MPI-1 versus MPI-2 communications and I/O generally show comparable results for configurations with less than 64 processors, but show 20-30% speedup for MPI-2 over MPI-1 for greater than 64 processors.

Conclusion

Through collaboration between the Naval Research Laboratory, Monterey and Lawrence Livermore National Laboratory, a portable, scalable version of the COAMPS atmospheric forecast model has been developed and adopted for numerical weather prediction. The model agrees with and maintains virtually all of the physical capability of the (now frozen) serial version. The capability of moving nests has been added. The model uses two-dimensional domain decomposition and handles parallelism through MPI with OpenMP. The model outperforms the Cray T90 and scales well to at least 50-100 processors.

Acknowledgements

Work performed under the auspices of the U.S.D.O.E. by University of California Lawrence Livermore National Laboratory under contract No. W-7405-ENG-48, and by the Naval Research Laboratory with support from the Office of Naval Research through Program PE-0602435N and from the Department of Defense through Program PE-0603755D.

References

Burk, S.D., and T. Haack, 2000: The dynamics of wave clouds upwind of coastal orography. *Mon. Wea. Rev.,* **128**, 1438-1455.

Dorman, C., T. Holt, D. Rogers, and K. Edwards, 2000: Large-scale structure of the June-July 1996 marine boundary layer along California and Oregon. *Mon. Wea. Rev*., **128**, 1632-1652.

Doyle, J.D., and M.A. Shapiro, 2000: A multi-scale simulation of an extreme downslope windstorm over Norway. Meteor. Atmos. Phys., **74**, 83-101.

Doyle, J.D., D.R. Durran, B.A. Colle, C. Chen, M. Georgelin, V. Grubisic, W.R. Hsu, C.Y. Huang, D. Landau, Y.L. Lin, G.S. Poulos, W.Y. Sun, D.B. Webe, M.G. Wuotele, and M. Xue, 2000: An intercomparison of model predicted wave breaking for the 11 Jan 1972 Boulder windstorm. *Mon. Wea. Rev.,* **128**, 901-914.

Haack, T., and S.D. Burk, 2001: Summer-time marine refractivity conditions along coastal California. J. Appl. Meteor., **40**, 673-688.

Haack, T., S.D. Burk, C. Dorman and D. Rogers, 2001: Supercritical flow interaction within the Cape Blanco-Cape Mendocino orographic complex. Mon. Wea. Rev., **129**, 688-708.

Hodur, R.M., 1997: The Naval Research Laboratory's Coupled Ocean/Atmosphere Mesoscale Prediction System (COAMPS), *Mon.Wea. Rev*., **125**, 1414-1430.

Liu, M., D.L. Westphal, T.R. Holt, and Q. Xu, 2000: Numerical simulation of a low-level jet over complex terrain in Southern Iran. *Mon. Wea. Rev*., **128**, 1309-1327.

Westphal, D.L., T.R. Holt, S.W. Chang, N.L. Baker, T.F. Hogan, L.R. Brody, R.A. Godfrey, J.S. Goerss, D.J. Laws, and C.W. Hines, 1999: A meteorological re-analysis for the study of Gulf War Illness. *Wea. Forecasting*., **14**, 215-241.

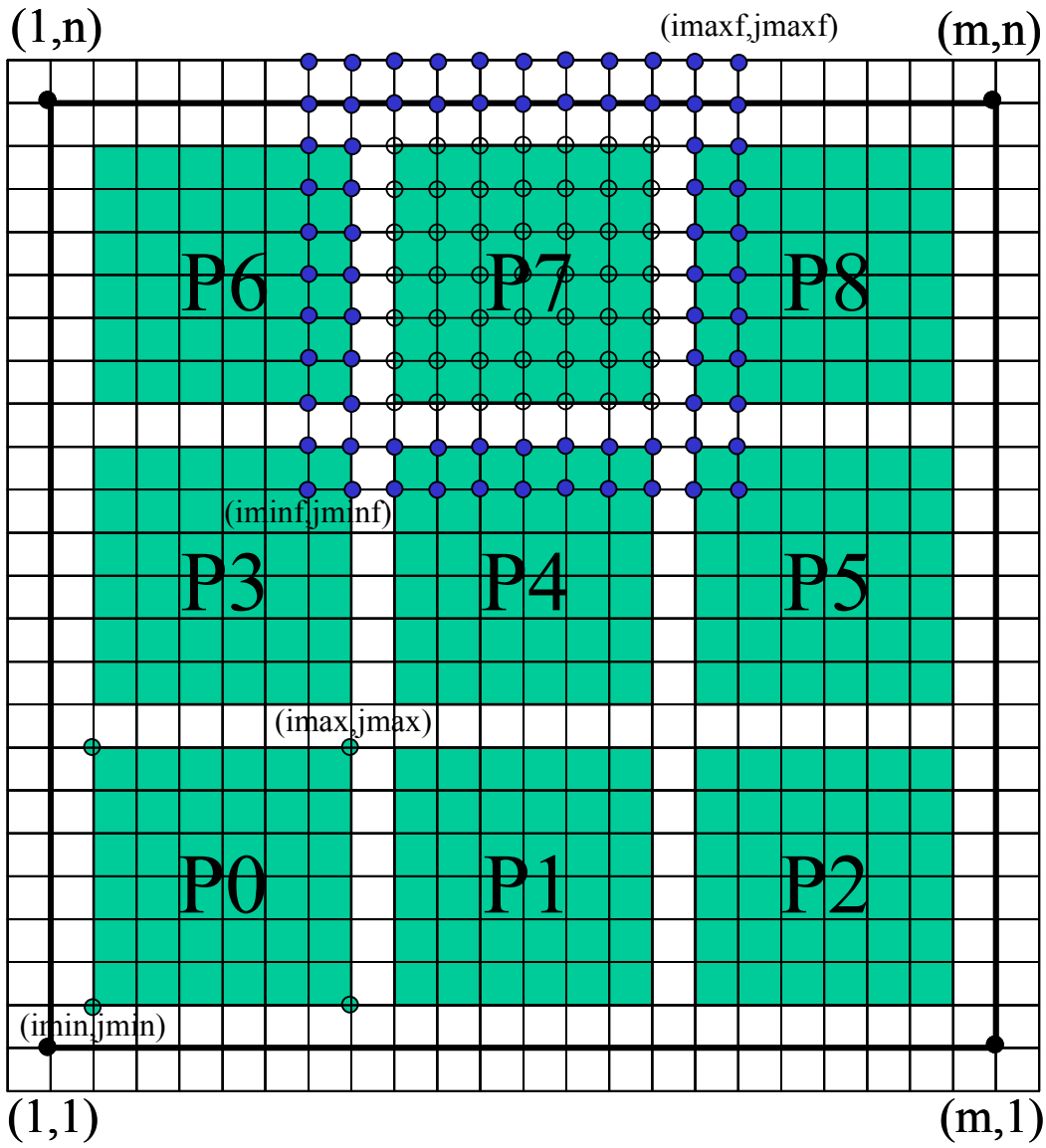

Figure 1. COAMPS single nest (m,n) 23 x 23 grid point domain for a 3 x 3 domain decomposition. The computational area of each subdomain in shaded. For subdomain 7 (P7) open circles indicate computational grid points and solid circles indicate the two halo points.

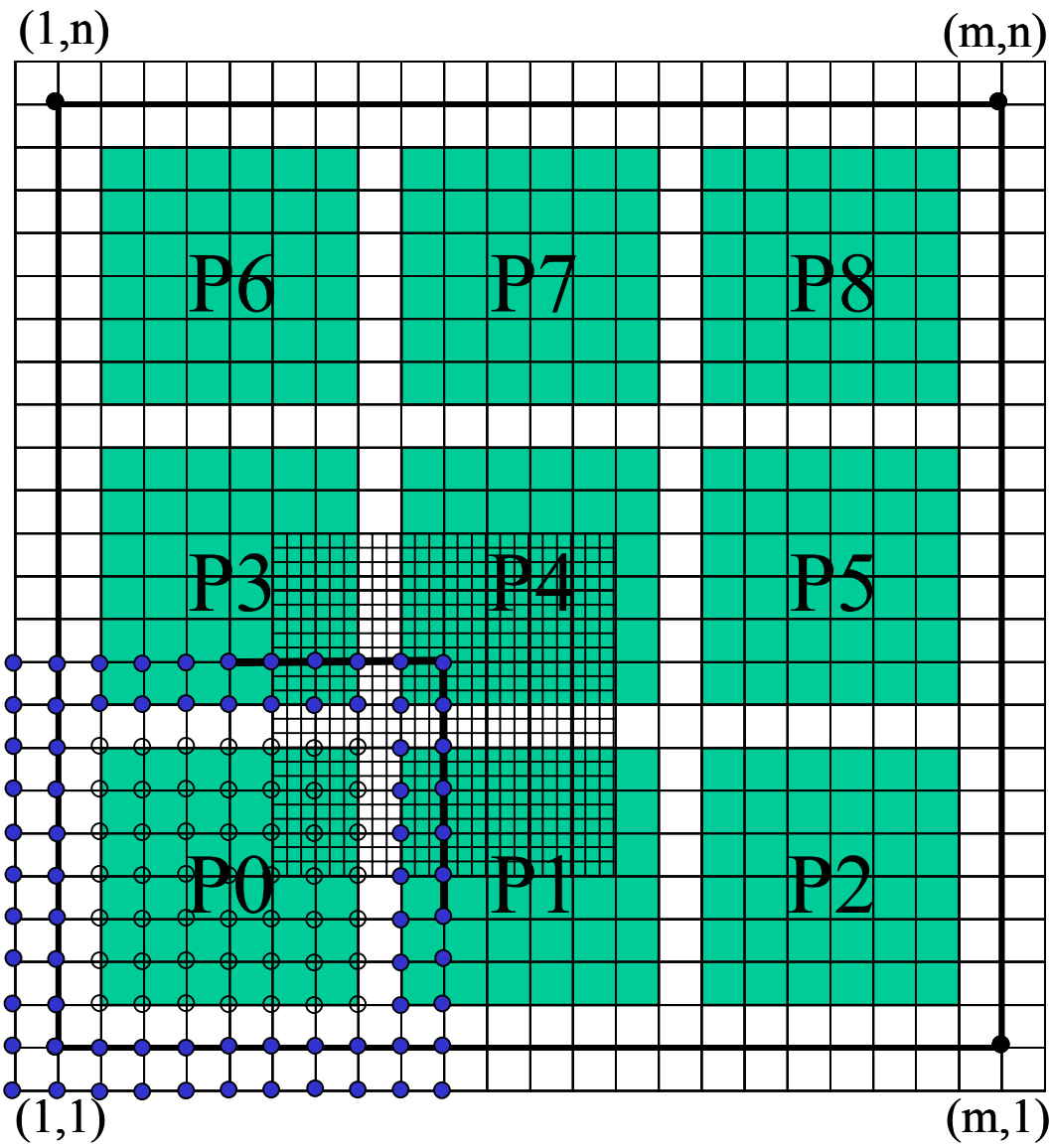

Figure 2. COAMPS parent nest domain similar to Figure 1, but in addition showing the overlap with a child nest. The open circles show the computational points and the closed circles show the two halo points of P0. The child nest is a 25 x 25 grid point domain. The heavy solid lines through the center of the child domain represent the outer extent of the parent subdomain for P0 that intersect with the child domain, as shown in Figure 3.

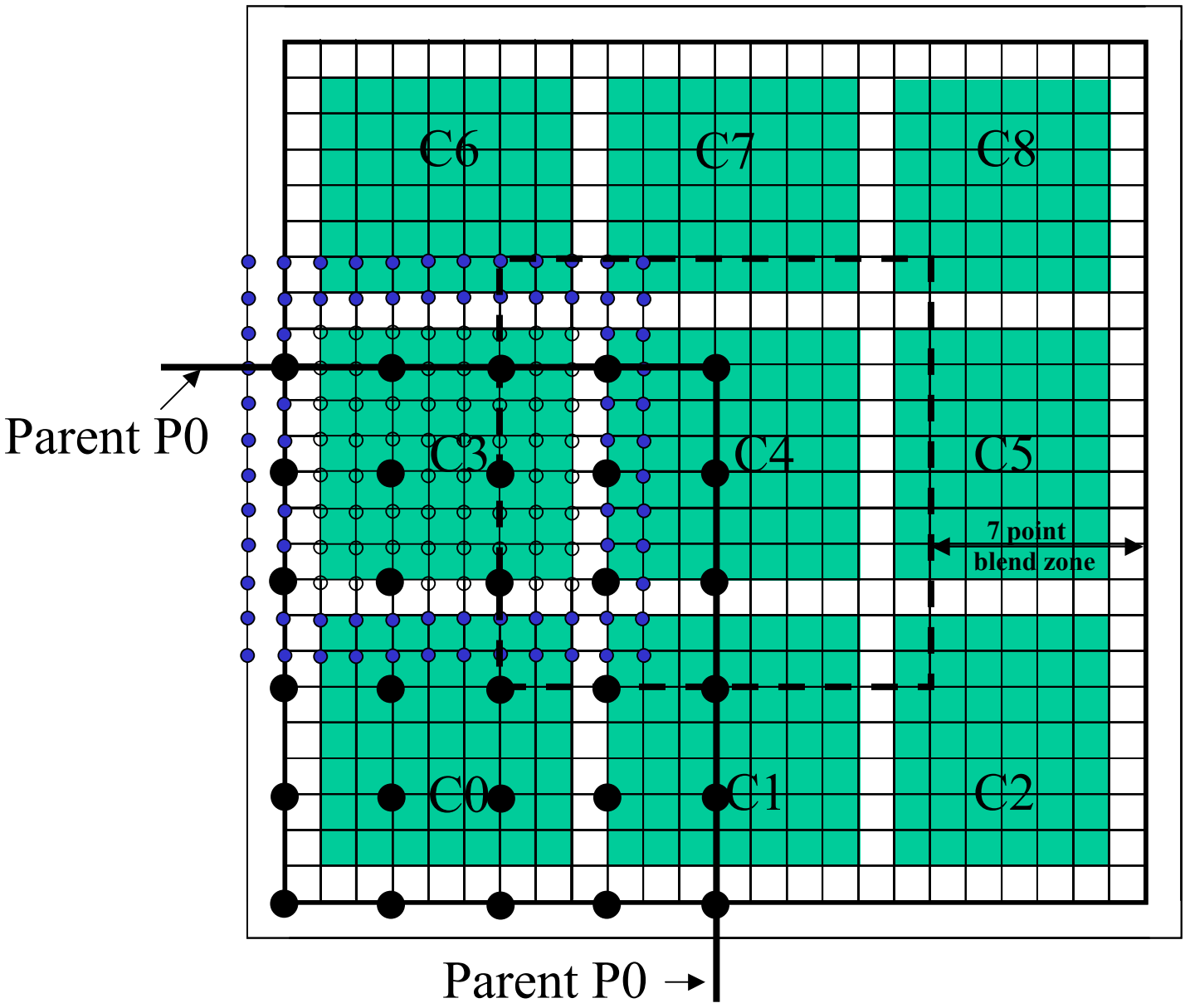

Figure 3. COAMPS child nest with 25 x 25 grid points and 3 x 3 domain decomposition. The heavy solid lines and larger solid circles represent the parent processor P0 as in Figure 2. The smaller open circles are the computational points and the closed circles are the two halo points of C3. The seven point blend zone is the area outside the dashed box.

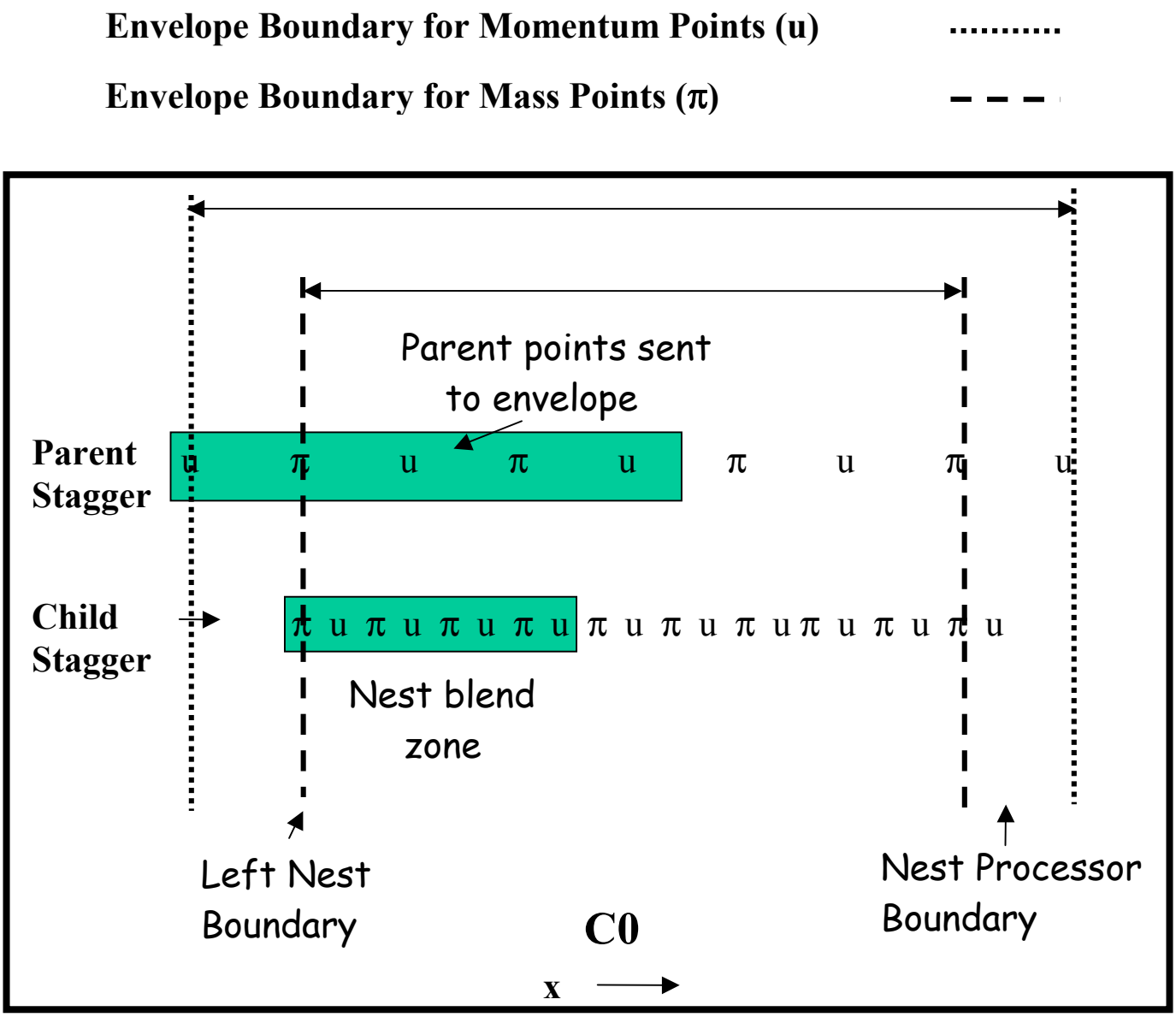

Figure 4. 1-D envelope structure for an Arakawa C-grid.

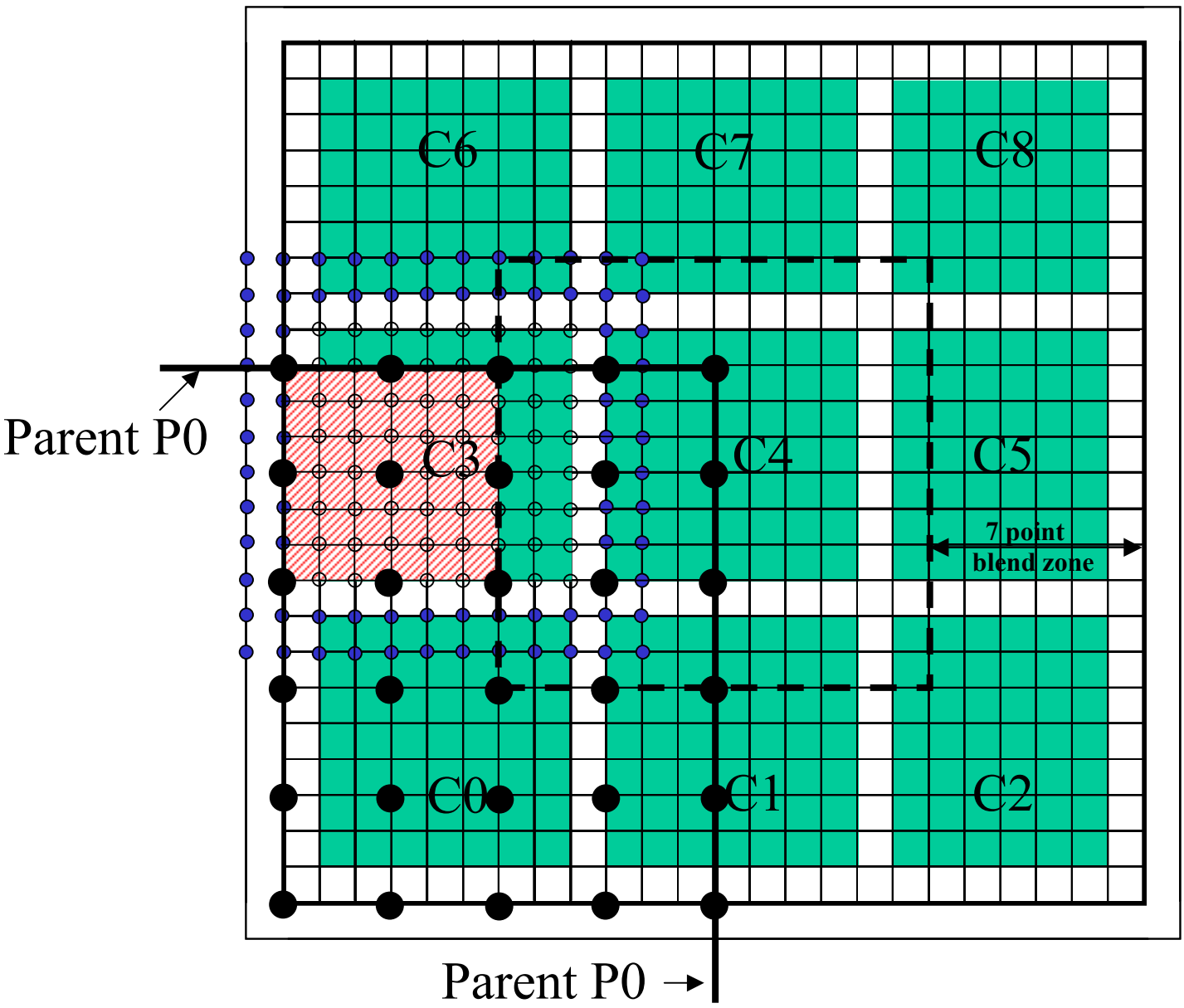

Figure 5. Sample COAMPS child nest and parent nest similar to Figure 3. The nine larger solid circles in the hatched region are the envelope of parent mass grid points that must be communicated from parent P0 to child C3 based upon the given seven point child blend zone. In this hatched region, mask values for parent and child nests are both equal to one.